BS21010 – programming report Group 4

**Introduction**

This is a three week project with the aim to make a program that is useful in a lab. The problem we decided to focus on is randomisation. Recently in our courses we have been studying statistics and in this module we have been discussing how to improve on experimental design. We have learnt that In a lab we aim to produce a bias free experiment whether it be conscious or unconscious and have as little error as possible or an equal error across are data set. One way to reduce the effect of error or random variation is to do multiple of treatments with a fair control so produced in multiples to. To do this we don’t want all the controls or treatments on the same day as a change in environmental factor may occur on that day. In order to stop this and stop the experimenter trying to bias treatments to favourable times the experimental order should be randomised.

We are attempting to wright a program that allows an experimenter to enter the number of controls and treatment they have with the number of days they have available and the number of treatments they can do in a day. This program then assigns the trials all names to return them in a randomised timetable through the graphical interphase with the possibility of providing blinding. The blinding proses will reveal a coded timetable to the experimenter and a decoding file to a blinder. There are other projects that do some similar aspects for instance (Python script for trial randomization by Erik Marsja | Oct 9, 2014 |), this project has a use of randomisation and a randomisation check to make sure no trials are consecutive with a dictionary- based output. These ideas we aimed to extend to our project. We want to improve on this with a graphical interface and a time interval given. We have written this program using python 3 in spider across git hub the program should typically used in a lab by an experiment.

The program has aims for an input of four sections one controls one ,of the treatments, one of the days available and one for the number of trials that can be ran in a day each with buttons that check there input and runs the program once all the inputs are turning correctly. The program will then assign random letter codes to each treatment and control, then randomises their position, from the new order formed they will be moved into dictionary of the number of trials per day for each day regardless of controls or treatment states. There will then be printed out as a time sheet inter-face with a file showing what each letter code stands for so if its being blinded the blinder can use these codes.

**Design**

To do this project we knew we would need some imported modules to do so. The first step of our code is to receive input and then randomise it to letters this requires the import of copy, random, string and from itertools permutations . This allows a string of letters to be permutated threw to randomly select a sequins of letter codes and copy them to a list. During test runs we realised that just 26 letters results in too low a repeat option for a scientific field. To remady this we increased are code to 3 letters . We also designed an interactive code so this needed the use of Tkinter for a GUI . We wanted a program that insures randomisation so our first step of design was to gave each trial a random code to make the output simple and unbiased then we built a randomiser to build into the time groups. We wanted a code that was interactive so that we could in sure the person could use it easily and not see the unwanted results so we built a graphical interface going through each of our inputs finally we designed a data output to show treatment for each day we designed this to show each day but not to specify days as experiment don’t always run Monday to Friday 9-5.

Randomise codes and place number per day of there in each dictionary till there full

Bild that many dictionary’s

User input to treatment panel

Build timetable from dictionary’s

Divide trial by trial by day

Bild file of trial and code

Check input button

User input are you the exsperimenter

For each trial assign log trial numbers of randomised letters

Integer input ?

**No yes**

**No**

User input Do you have a blinder

**Yes yes**

Take log of trials

Return: correct input

**No**

Save file

Valid input

User input to control panel

**Yes**

input ok

**No**

Check input button

Integer input

**No**

Check input button

User input trails per day

**Yes**

Return: correct input

Trials <17576

Return: input ok

**Yes**

**No**

Return you have to many trials

We began with the building of a base code to do each of our aims and then split this into functions to then build across to the interface. We wanted to build in error checking so we added cheeks to all our in inputs in the form of the input buttons. The design of are letter codes was chosen to allow blinding as the output is automatically blinded and a file is used to change this we also chose to double randomise as an extra step to insure its far enuf away from there starting order. We added an error checking for human error in the time inputs so if there is not enough time for the exsperiemt to be run the user is woned and if there is too much time it is ignored this improved design as before if the time evalabul was insafishent dat wa simpaly ignored.

**Code description**

The code can be found at <https://github.com/SerGe0rge/Group-project-BS21010> . The functions used are

#1) codegenerator(): this generate a list of distinct 3 letter codes for each control and treatment

#2) assigncodes(): in this function codes are assigned to controls(lowercase) and treatments(uppercase) in list there cases are when unblinded to make it easier when creating an unblinded file

#3) tabulateblind(): the function insures treatment list codes are then all changed to lower case so it cannot be differentiated which is a control code and which is a treatment code. This is displayed as an output since all codes are in lowercase.this is the unblided timetable. The option to Save is provided.

#4) tabulateunblind(): this function can be called to create a unblinded table which will not be displayed as an output to ensure there is no bias if a scientist conducting the experiment is using this program. a save option is provided. the saved file can be opened and viewed by the one not conducting the experiment to alow exsperimenas set up and analysis.

set\_controls()

set\_treatments()

set\_days()

set\_numexpt()

cd\_gen()

# dis\_cd()

# table()

root.mainloop()

the first functions where built into the class to alow

This program should be incorporated into experiments during the design phase of an experiment once the number of trials and repeats has been decided and the first small test experiment has been run to show the technique is valid and to discover the time the experiments take the experimenter can pass there information for there main experiment to the program and check them to see if there compatible to then receive there timetable. The blinder can then get there unblinded version and take the unblinded timetable and set up the lab proses with the blinding codes. Once the experiment is complete and the report is being written the files can be used to unblinded the results for analysis.

**Conclusion**

Dew to time constraints we could not do all we wished to with this program for instance in our program there is a limit to the number of trials that can be randomised, only 17676 can be inputted as this is the maximum number of possible letter codes possible because the code is limited to three letters and for three sets of the alphabet there is only 17676 possible individual codes. In science some trials can have millions of trials needed to be successful so this could be a possible improvement. Another issue with are code is that there is no way to name each of your controls and treatments so if you need to tell them apart, for instance you have multiple trials of one type and some of another you would have to take a record of how many you have and get all the codes from the unblinded file this may be to much work in large experiments and makes blinding far harder. we have not added a checker to insure that there are no repeats so we are running the code on the assumption the permutations run secsesfuly with more time it could be built with a ckeker and more options of carecters to extend the variation. The code also dose not allow for latter alterations without creating a hole new trial order, so if the experimenter did there first day of experimentation and discovered they where getting less trials done than they expecting they cannot ask the code to adapt there current timetable it can only make an Entirely new one wasting the first days worth of experimental data.